Universidad Autónoma de Madrid

Escuela Politécnica Superior

Grado en Ingeniería Informática

TRABAJO DE FIN DE GRADO

ARDUCRIPTO: LIBRERÍA CRIPTOGRÁFICA PARA MICROCONTROLADORES DE BAJO COSTE

Julián Isaac Ojito Herrero
Tutor: Oscar Delgado Mohatar
Ponente: Eloy Anguiano Rey

Julio 2016

ARDUCRIPTO: LIBRERÍA CRIPTOGRÁFICA PARA MICROCONTROLADORES DE BAJO COSTE

Autor: Julián Isaac Ojito Herrero
Tutor: Oscar Delgado Mohatar
Ponente: Eloy Anguiano Rey

Departamento de Software
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Julio 2016

Agradecimientos

Me gustaría expresar mi agradecimiento a todas aquellas personas que con su ayuda han colaborado y ayudado en la realización del presente trabajo.

En primer lugar, a mi tutor Oscar, por confiar en mí y ofrecerme un TFG de un tema interesante y actual.

En segundo lugar, a mis compañeros de clase, a los villajujeros, compañeros de armas del Conter Strike o compañeros de troleadas en el Lol y, de manera especial, a Guido el salvaje, que siempre tenía alguna palabra para sacarme una sonrisa. También a Sergio y Ana, la pareja oficial de la EPS, a Cristina, que sin ella este trabajo no se hubiese terminado y a Jose, una persona a la que considero un gran amigo y por la que repetiría encantado la carrera si supiese que volvería a conocerle.

Por último, pero no por ello menos importante, a mi familia, a mi padre por no dejar nunca de tirarme de las orejas si hacía falta, a mi madre por tener siempre su confianza y a mi hermana, que por suerte para mí es un ejemplo a seguir.

Abstract

Abstract — We live in a time of continuous technological progress, in which the importance of maintaining the security of both corporate data and the information which is transmitted between them is vital. This is one of the main reasons why fields such as cyber security and cryptography are becoming a major force in the market and a large labor demand, due to the need by corporations to be updated in the latest defense mechanisms regarding data protection.

Coupled with this, the use of microcontrollers for all kinds of tasks has suffered a large increase because of its low cost and its portability, allowing to cheapen and facilitate many of the tasks of the corporations.

It is for this reason that this project attempts to unite these two fields, first cybersecurity and on the other hand microcontrollers, in an attempt to obtain a good performance of cryptographic algorithms, and to ensure a high level of security for the data, transmited and kept localy. In the preparation of this work, there have been implemented different cryptographic algorithms widely used such as AES, HASH or ECC, all implemented on an Arduino Mega as it is a microcontroller open source, which facilitates their preparation and their further development as well as having less capacity than other microcontrollers as Raspberry, since it is intended to use the least possible resources.

In short, the objectives of this work are to check whether it is possible to implement and operate correctly some of the cryptographic algorithms most used and detailed above, obtaining relatively reasonable times so that it can be implemented in a real application and using the least necessary resources.

Key words — Arduino Mega, Cybersecurity, Cryptography, Microcontroller.

Resumen

Resumen — Vivimos en una época de continuo avance tecnológico, en el que la importancia de mantener seguros tanto los datos de las empresas como la información que se transmite entre ellas es vital. Este es uno de los principales motivos por los que campos como la ciberseguridad y la criptografía están cobrando una gran fuerza en el mercado, así como una gran demanda laboral debido a la necesidad por parte de las corporaciones de estar a la última en cuanto a mecanismos de defensa a la hora de proteger los datos, tanto suyos como de sus clientes.

Unido a esto, la utilización de microcontroladores para realizar todo tipo de tareas ha sufrido un gran aumento debido a su bajo coste, en comparación con el equipamiento tradicional utilizado, y a su portabilidad, lo que ha permitido abaratar y facilitar muchas de las labores de las corporaciones.

Es por este motivo que en este proyecto se intenta unir estos dos campos, por un lado la ciberseguridad y por otro lado los microcontroladores, en un intento por obtener un buen rendimiento de los algoritmos criptográficos, así como para garantizar un alto nivel de seguridad en cuanto a la protección de los datos, tanto estáticos como en tránsito.

En la elaboración de este trabajo se han implementado diferentes algoritmos criptográficos muy utilizados como son AES, HASH o ECC, todos ellos implementados en una placa Arduino Mega, ya que se trata de un microcontrolador open source, lo que facilita su obtención y su posterior desarrollo, además de tener menos capacidad que otros microcontroladores como Raspberry, ya que se pretende utilizar los menores recursos posibles. En definitiva, los objetivos de este trabajo son comprobar si se pueden implementar y hacer funcionar de manera correcta algunos de los algoritmos criptográficos más utilizados y detallados anteriormente, obteniendo tiempos relativamente razonables para que su implantación en una aplicación real pueda ser factible y utilizando los menores recursos necesarios.

Palabras clave — Arduino Mega, Ciberseguridad, Criptografía, Microcontrolador.

Glosario

Diecimila El Arduino Diecimila es una placa electrónica basada en el ATmega168. Cuenta con 14 pines digitales de entrada / salida (de los cuales 6 se podrán utilizar como salidas PWM), 6 entradas analógicas, un 16 MHz oscilador de cristal, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o la corriente con un adaptador de CA a CC o una batería para empezar. 17

Plug and Play Característica de aquellos dispositivos que pueden utilizarse en una computadora sin la necesidad de configuración. 8

Acrónimos

AES Advanced Encryption Standard. 4, 10, 13–16, 23, 30, 36, 37

CBC Cipher Block Chaining. 9

CFB Cipher Feedback. 9

COM Communications. 21, 22

DES Data Encryption Standard. 10

DSA Digital Signature Algorithm. 38

ECB Electronic Codebook. 9

ECC Elliptic Curve Cryptography. 4, 14, 15, 23, 24, 30–33, 37, 38

ECDH Elliptic Curve Diffie-Hellman. 11, 13, 15, 30, 31

FTDI Future Technology Devices International Ltd. 9

GND Ground. 8

ICSP In-Circuit Serial Programming. 17

IVREA Interaction Design Institute Ivrea (Italy). 7

M2M Machine-to-Machine. 2

MCU Microcontroller Unit. 1–3, 7, 17, 22

OFB Output Feedback. 9

PC Personal Computer. 4, 35

PWM Pulse Width Modulation. 9, 17, 18

RSA Rivest Shamir Adleman. 4, 11, 37, 38

SHA Secure Hash Algorithm. 4, 11, 13, 14, 30, 36, 37

TDP Thermal Design Power. 18

TEA Tiny Encryption Algorithm. 10

TI Tecnología de la información. 1

TTL Transistor-Transistor Logic. 9

UART Universal Asynchronous Receiver-Transmitter. 17

USB Universal Serial Bus. 8, 17

XXTEA Corrected Block TEA. 4, 10, 13, 16, 24, 30, 36, 37

Índice general

Índice de figuras

Índice de tablas

1 Introducción

En este capítulo se expondrán los motivos que han llevado a la realización de este proyecto, las fases en las que se ha dividido, los objetivos finales del proyecto y por último se detallará la estructura del resto del documento.

1.1. Motivación

En la actualidad, los ciber ataques son cada vez más comunes y sus consecuencias pueden afectar a empresas, gobiernos, personas que utilizan diariamente sus dispositivos o, incluso, pueden llegar a afectar a sectores claves de un país. El rápido avance de la tecnología, la gran interconectividad de los dispositivos tecnológicos y la falta de madurez en la industria de Tecnología de la información (TI), hace que las posibilidades para los posibles atacantes sean aún mayores. Por los motivos expuestos anteriormente, es necesario conocer las posibles amenazas para poder anticiparse a los movimientos de los atacantes y controlar los posibles ataques que realicen. Es en este punto donde toma protagonismo y adquiere una gran importancia la ciberseguridad.

La ciberseguridad y los MCU han tomado gran importancia durante los últimos años. La ciberseguridad es un conjunto de herramientas, políticas, conceptos de seguridad, directrices, métodos de gestión de riesgos, acciones, prácticas y tecnologías que son utilizadas para proteger a los usuarios de las amenazas que ponen en riesgo la información que se almacena, se procesa, o se transmite, reduciendo al mínimo los riesgos. [1]

El uso de los MCU en el campo de la ciberseguridad durante los últimos años ha ido en aumento. Según el informe "Embedded Digital Security World - 2013", el uso de MCU en la seguridad (MCU para ciberseguridad) se incrementará en 529 millones de unidades

en 2017, lo que supondrá un incremento del 92 % desde el año 2013, ver Figura 1.1 [2].

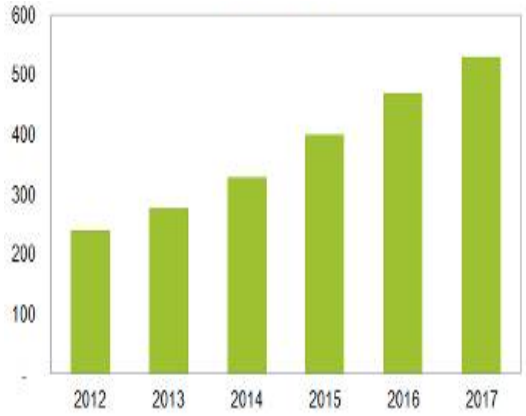

Figura 1.1: Evolución de la previsión de uso de MCU en seguridad (en millones). (Fuente: [3])

En la Figura 1.1 se puede ver el pronóstico de uso de los MCU en el campo de la seguridad. Estos MCU se usan principalmente en tres aplicaciones: módulos de plataforma segura, autenticación y Machine-to-Machine (M2M).

El incremento del uso de los MCU se debe a la reducción de costes, de energía dentro de un sistema y al tamaño tan reducido que poseen estas placas. Algunos de los MCU más usados son los que se muestran en la Tabla 1.1.

Empresa	8 bits	16 bits	32 bits
Atmel	AVR, 89xxxx familia similar 8051	-	SAM7, SAM3, SAM9, AVR32
Freescale	68HC05, 68HC08, 68HC11, HCS08	68HC12, 68HC12, 68HCSX12, 68HC16	683xx, PowerPC, ColdFire
Holtek	HT8	-	-
Intel	MCS-48, MCS51, 8xC251	MCS96, MXS296	x
National Semiconductor	Cop8	x	x
Microchip	Familias 10f2xx, 12Cxx, 12Fxx, 16Cxx, 16Fxx, 18Cxx y 18Fxx	PIC24F, Pic24H, dsPIC30Fxx, dsPIC33F	PIC32
NXP Semiconductors	80C51	XA	Cortex-M3, Cortex-M0, ARM7, ARM9
Renesas	78K, H8	H8S, 78K0R, R8C, R32C/M32C/M16C	RX, V850, SuperH, SH-Mobile, H8SX
STMicroelectronics	ST 62, ST 7	-	STM32

Empresa	8 bits	16 bits	32 bits
Texas Instruments	TMS370	MSP430	C2000, Cortex-M3, TMS570
Zilog	Z8, Z86E02	-	-

Tabla 1.1: MCU más usados.

La empresa Atmel es una de las empresas más importantes en la producción de MCU para Arduino, e invierte grandes cantidades de dinero en la producción de estos dispositivos. Es por esto, que se ha utilizado una placa Arduino con un MCU de esta empresa.

Es importante resaltar que estos MCU pueden usarse en millones de campos. En la Figura 1.2 se recogen algunos ejemplos de estos usos. [4]

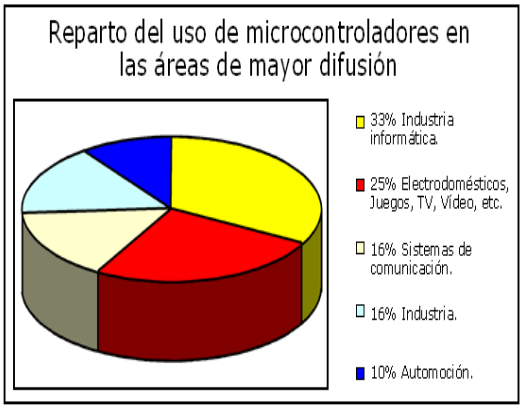

Figura 1.2: Uso de MCU en diferentes áreas. (Fuente: [4])

1.2. Objetivos

Como ya se dijo en el apartado anterior, la ciberseguridad ha adquirido una gran importancia en esta era de la tecnología. Una forma de obtener seguridad en este mundo es a través de algoritmos criptográficos.

La implementación de algoritmos criptográficos se ha realizado en diferentes lenguajes y en equipos con gran capacidad computacional y memoria, donde no importa el tamaño del programa implementado o el tiempo computacional que este programa utilice. Sin embargo, Arduino posee recursos limitados, como por ejemplo la memoria, lo que ha hecho que sea un reto implementar estos algoritmos en un sistema con estas características.

Dicho esto, los principales objetivos de este trabajo son:

- Implementación de las principales primitivas criptográficas para MCU de baja capacidad computacional y bajo coste, incluyendo cifrado, funciones hash e intercambio de claves (Diffie-Hellman).

- Benchmarking y caracterización de estas primitivas, comparándolas con las mismas implementaciones en un entorno Personal Computer (PC) de sobremesa.

- Determinación de la posibilidad de la ejecución de criptografía de clave pública tradicional, basada en la factorización de enteros (algoritmo Rivest Shamir Adleman (RSA)). Si no es posible, implementación de algoritmos basados en curvas elípticas (ECC).

En este proyecto se va a abordar la implementación de algunos algoritmos criptográficos en una placa de poca capacidad como es Arduino. Se desea comprobar si los algoritmos criptográficos pueden implementarse en estos dispositivos, si el funcionamiento de los algoritmos es el correcto y el rendimiento que se obtiene tras su ejecución.

1.3. Estructura del documento

La realización de este trabajo se ha dividido en diferentes capítulos:

- **Capítulo 1**: para comenzar, se abordarán las razones que han motivado a la realización de este proyecto y los objetivos que se establecieron al iniciarlo.

- **Capítulo 2**: en este capítulo se detallarán las partes que conforman este proyecto, los orígenes de Arduino y sus utilidades, su gran crecimiento en los últimos años y también se hablará de las grandes y diferentes utilidades que tiene en la sociedad. Por último, se abordará el tema de la ciberseguridad, se expondrá la gran importancia que tiene en la actualidad, donde todos los equipos pueden estar conectados a través de internet, concretamente se profundizará en la seguridad que aportan los algoritmos criptográficos.

- **Capítulo 3**: en el tercer capítulo se detallará el diseño establecido para el desarrollo de las diferentes librerías implementadas, las cuales implementan los algoritmos criptográficos XXTEA, SHA, AES y ECC. También se muestra el flujo de comunicación entre el ordenador y la placa Arduino, a través del puerto serie, para cada uno de los algoritmos criptográficos implementados.

- **Capítulo 4**: en él se listará el hardware utilizado, el entorno en el que se ha realizado este trabajo y los diferentes cambios que se han tenido que realizar en los programas para poder obtener un correcto funcionamiento. Se hablará de las principales funciones de cada una de las librerías implementas, explicando que funciones realizan y que parámetros necesitan para efectuar las operaciones.

- **Capítulo 5**: se explicarán las pruebas que se han realizado en el proyecto. Con esta batería de pruebas se verifica el correcto funcionamiento de las librerías de Arduino, es decir, que la implementación de estos algoritmos criptográficos ha sido correcta y genera las salidas esperadas.

- **Capítulo 6**: por último, se muestra el funcionamiento de los algoritmos criptográficos en Arduino y en Linux. Se mostrarán, en ambos casos, los tiempos obtenidos tras la ejecución de los diversos algoritmos. Para finalizar el capítulo, se expondrá el posible trabajo a realizar en un fututo una vez concluido este proyecto.

2

Estado del Arte

A día de hoy, el campo de la robótica está en auge y, dentro de este mundo de cables y placas, uno de los dispositivo electrónicos más utilizados es la placa Arduino. Arduino es una plataforma hardware de código abierto basada en una sencilla placa con entradas y salidas, analógicas y digitales, desarrollada en un entorno de programación que posee un lenguaje Processing. Este dispositivo contiene en su interior las tres principales partes de un ordenador, una unidad central de procesamiento, una memoria y la posibilidad de conectar periféricos tanto de entrada como de salida, y es capaz de conectar el mundo físico con el mundo virtual y el mundo analógico con el digital.

2.1. Arduino

Arduino se inició en el año 2006 como un proyecto para estudiantes en el Instituto Interaction Design Institute Ivrea (Italy) (IVREA), donde recibió el nombre de Arduino por el Bar di Re Arduino (Bar del Rey Arduino). Hasta ese momento, se usaban los MCU BASIC Stamp, los cuales eran considerados demasiado costosos debido a su elevado precio, 100 dólares estadounidenses. [3]

Hernando Barragán, estudiante colombiano, colaboró en el desarrollo del proyecto, desarrollando las tarjetas electrónicas Wiring, su lenguaje de programación y la plataforma de desarrollo. Una vez concluida la plataforma de desarrollo, los investigadores trabajaron en el abaratamiento de las placas Arduino, la reducción de su peso y su tamaño y, por último, en la posibilidad de hacerlo accesible, de manera que se permitiese que la placa fuese accesible para la comunidad de código abierto (hardware y código abierto). [3]

Años más tarde, Google realizó una colaboración con este grupo de estudiantes para

desarrollar el Kit Android ADK (Accesory Development Kit). El objetivo del proyecto era desarrollar una placa Arduino capaz de comunicarse directamente con teléfonos móviles inteligentes bajo el sistema operativo Android. Se deseaba que la placa permitiese al teléfono controlar luces, motores y sensores conectados a ella. [3]

Para la producción serie de la placa, se tuvo en cuenta que el coste no superase los 30 euros, que la placa fuese de color azul, Plug and Play y que trabajase con todas las plataformas informáticas, como por ejemplo MacOSX, Windows o GNU/Linux. [3]

En la actualidad el uso de Arduino se aplica en infinidad de proyectos, en ámbitos como la educación o la domótica. Algunos de estos proyectos son los siguientes:

- **Quakescape 3D**: es un proyecto que permite comprobar la intensidad que tendría un terremoto real a partir del uso de una maqueta y pintura. De esta manera se podría simular un terremoto y ver qué consecuencias tendría. [5]
- **Eyewrite 2.0**: este proyecto permite dibujar utilizando los ojos, de manera que personas que tienen discapacidad motora puedan dibujar con el simple movimiento de sus ojos. [6]
- **Turn signal biking jacket**: es un proyecto que, mediante el uso Arduino LilyPad y unos leds, permite a los ciclistas indicar la dirección que van a tomar. Con el movimiento de los brazos, los leds que llevan incorporados en el chaleco se iluminan indicando la dirección que van a tomar. [7]

Una vez expuesto el origen de Arduino, se explicarán las características generales de todas las placas Arduino, y se explicarán los algoritmos criptográficos implementados en este trabajo.

2.2. Características generales

Todas las placas Arduino disponen de: [8]

- **Conexión fuente de alimentación**: usada para alimentar la placa, ya sea con un transformador o una pila de 9V.
- **Conexión Universal Serial Bus (USB)**: usada para alimentar la placa.
- **Pin reset**: usado para resetear la placa.
- **Pin de referencia analógico**: usado como referencia de voltaje para entradas analógicas.
- **2 pines Ground (GND)**: tomas de tierra.
- **Pines digitales**: usados para enviar/recibir señales digitales de 0 y 5V.

- **Pines analógicos**: las entradas se usan para obtener datos de sensores, las salidas se utilizan para enviar señales Pulse Width Modulation (PWM).
- **Pin/pines salida serie TX**: usados para trasmisiones series de señales Transistor-Transistor Logic (TTL).
- **Pin/pines entrada serie RX**: usados para trasmisiones series de señales TTL.
- **Programador serie**: usado para establecer conexión serie.
- **Pin Vin**: usado para aplicarle la tensión de entrada.
- **Pin 3v3**: genera 3,3 voltios por el chip Future Technology Devices International Ltd (FTDI).
- **Pin 5v**: se usa para alimentar la placa y otros componentes de la placa.

2.3. Criptografía

La criptografía es el conjunto de técnicas utilizadas para ocultar o proteger la comunicación, la información y a las personas que se comunican, de personas no autorizadas. La criptografía se clasifica en dos grandes grupos, criptografía simétrica y criptografía asimétrica.[9]

Criptografía simétrica

Se consideran sistemas de criptografía simétrica todos aquellos sistemas criptográficos que cifran y descifran con la misma clave, la cual debe ser conocida con anterioridad por el emisor y el receptor. El punto débil de estos sistemas es la transferencia de esta clave entre los dos puntos de la comunicación. [9]

Cifrado por bloques

El cifrado por bloque es una unidad de cifrado de clave simétrica que trabaja con porciones del texto de tamaño fijo que, a través de permutaciones y sustituciones, obtiene el texto de tamaño fijo cifrado. El modo de operar con los bloques de texto es lo que se denomina modo de cifrado. Algunos de los modos de cifrado más importantes son, Electronic Codebook (ECB), Cipher Block Chaining (CBC), Output Feedback (OFB) o Cipher Feedback (CFB). En este caso, se ha implementado el modo CBC que, antes de cifrar un bloque, realiza la operación xor entre un bloque de texto claro y el anterior bloque de texto cifrado. En la práctica es el método más usado. [9]

- **AES**: AES es un algoritmo que remplazó al algoritmo de clave simétrica Data Encryption Standard (DES). Este algoritmo utiliza bloques de datos de 128 bits y tres tamaños diferentes de clave, 128 bits, 192 bits, y 256 bits. En este proyecto se ha implementado el algoritmo AES de 256 bits. Este algoritmo opera con una matriz de 4x4 denominada State. Las operaciones básicas que realiza el algoritmos son AddRoundKey, SubByte, ShiftRows, MixColumns, y por último de Key Schedule. [10] En la Figura 2.1 se puede ver el funcionamiento del algoritmo.

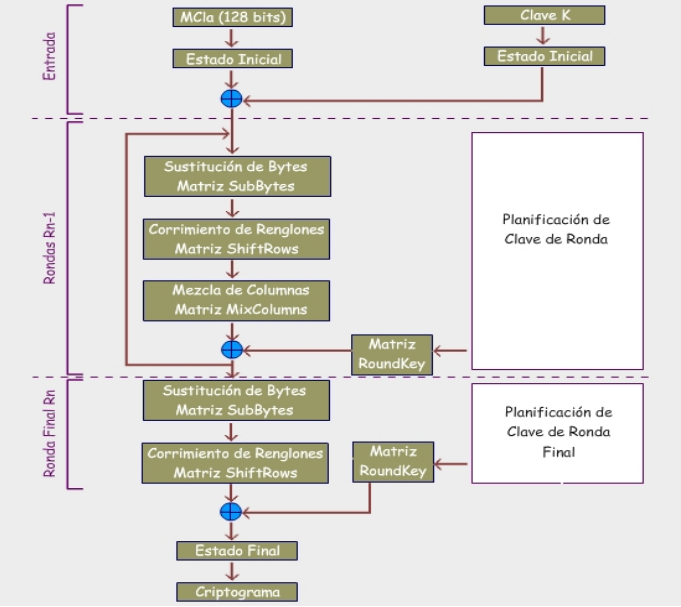

Figura 2.1: Funcionamiento del algoritmo AES. (Fuente: [3])

- **XXTEA**: XXTEA es un algoritmo de cifrado por bloques que se implementó con el objetivo de eliminar las vulnerabilidades de Tiny Encryption Algorithm (TEA). XXTEA está basado en la red Feitsel, y opera con al menos dos bloques de 32 bits y una clave de 128 bits. El número de ciclos que realiza el algoritmo depende del tamaño del bloque. [11]

Criptografía asimétrica

La criptografía asimétrica es aquella que usa dos claves, una pública y otra privada. La clave pública se puede difundir o dar a conocer a cualquier persona, sin embargo, la privada nunca se debería entregar o mostrar. En este tipo de criptografía, para cifrar un mensaje se utilizaría la clave pública del destinatario, de esta manera solo ese usuario podría descifrar el mensaje con su clave privada. [9]

Hash

Hash es un algoritmo que genera una salida alfanumérica, normalmente de longitud fija, a partir de una entrada. Es una función inyectiva, es decir, que a cada valor lo representa una única salida, uniforme, y de bajo costo computacional. Las funciones hash implementadas han sido, SHA-256 y SHA-512, ambas pertenecen al conjunto de funciones hash denominado SHA-2. [10] Anterior a este, estaba el SHA-1, el cual se demostró que tenía fallos de seguridad, sin embargo aun compartiendo el funcionamiento con SHA-2, en este segundo caso los ataques realizados no han sido satisfactorios. En Figura 2.2 se puede observar una pequeña comparación de las dos funciones hash implementadas. [12]

Algoritmo	Tam.Salida	Tam.Bloque	Tam. Máximo del mensaje	Log.Cadena	Iteraciones	Operaciones	Colisiones
SHA -256	256	256	$2^{64} - 1$	32	64	+,and,or,xor,shr, rot	Ninguna
SHA -512	512	512	$2^{128} - 1$	64	80	+,and,or,xor,shr, rot	Ninguna

Figura 2.2: Comparación de SHA-256 y SHA-512.

Curvas Elípticas

Las curvas elípticas aparecen como la gran alternativa a los criptositemas de clave pública como RSA, ya que consiguen una disminución del tamaño de las claves y un nivel de seguridad equivalente. En este proyecto se ha realizado la implementación del algoritmo Elliptic Curve Diffie-Hellman (ECDH) de intercambio de claves de Diffie-Hellman que, en realidad, más que un algoritmo de cifrado es un protocolo para establecer una clave simétrica. Aplicando este algoritmo, se obtiene una comunicación segura entre dos puntos (a y b por ejemplo). [13]

El modo de ejecución seria el siguiente:

1. Los extremos a y b generan su clave pública y su clave privada.
2. El extremo a envía su clave pública al extremo b.
3. El extremo b envía su clave pública al extremo a.
4. Tanto a como b, calculan la clave simétrica usando su clave privada y la clave pública del otro extremo.

De esta manera, si hubiese alguien escuchando en el medio solo tendría la clave pública de ambos puntos impidiéndole, de esta manera, que pueda llegar a conseguir la clave simétrica generada por ambos puntos.

3 Diseño

En esta sección se describirá el diseño de las librerías confeccionadas y de los algoritmos criptográficos implementados. Se comenzará listando las librerías que se han implementado y posteriormente se explicará el flujo de cada uno de los algoritmos contenidos en las librerías.

3.1. Diseño de librerías

Cuando hablamos de las librerías, nos referimos al conjunto de sentencias y funciones que se han implementado para abrir conexiones entre Arduino y los elementos que se conectan a él. Estas librerías no se encuentran dentro de las librerías básicas de la IDE y en este proyecto tienen una gran importancia. Las librerías implementadas son las siguientes:

- **ShArduino**: librería que implementa los algoritmos hash SHA-256 y SHA-512.
- **AesArduino**: librería que implementa el algoritmos simétrico AES-256.
- **XXTeaArduino**: librería que implementa el algoritmo simétrico XXTEA.
- **EccArduino**: librería que implementa el algoritmo ECDH.

3.2. Flujo del sistema

A continuación, se explicará el flujo de la comunicación en cada uno de los algoritmos. En esta comunicación intervienen hasta un máximo de dos placas Arduino, en el caso del algoritmo ECC, y un ordenador, que será el encargado iniciar la comunicación con las placas Arduino.

3.2.1. SHA-2

En la implementación del algoritmo hash intervienen un ordenador y un Arduino. Esta comunicación sigue un esquema como el que se define a continuación:

- Desde el ordenador se envían a Arduino los datos que se quiere que formen el hash, ya sea el 256 o el 512.
- Arduino genera el hash con los datos recibidos.
- Arduino envía el hash realizado al ordenador.

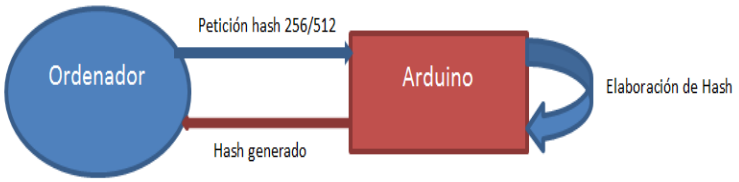

Figura 3.1: Diagrama de flujo de SHA-256 y SHA-512.

3.2.2. AES-128

En el caso del algoritmo simétrico AES-256, al igual que en el algoritmo hash, intervienen un ordenador y una placa Arduino. El modelo que implementa es el siguiente:

- El ordenador envía a la placa Arduino la petición de cifrar/descifrar junto al mensaje.
- En función de la orden recibida del ordenador, Arduino realiza el cifrado/descifrado del mensaje que ha recibido.
- Arduino envía al ordenador el resultado del cifrado/descifrado del mensaje.

Los algoritmos de encriptación por bloques cumplen el esquema que se muestra en la Figura 3.2.

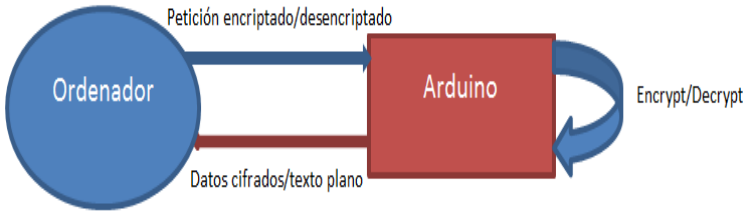

Figura 3.2: Diagrama de flujo de AES-128.

3.2.3. ECC

Como se comentó en el apartado anterior, el algoritmo de curva elíptica implementado es el ECDH de intercambio de claves de Diffie-Hellman. En su ejecución, se han utilizado dos placas Arduino y un ordenador. Finalmente, el modelo que sigue este algoritmo es el que se detalla a continuación:

- El ordenador envía una petición a las placas Arduino para generar las claves públicas.
- Las dos placas Arduino generan sus claves públicas.
- Cada una de las placas envía su clave pública al ordenador.
- El ordenador intercambia esas claves públicas con las placas.
- Una vez recibida la clave pública de la otra placa Arduino, estas generan la clave simétrica.
- Cada Arduino envía su clave simétrica generada al ordenador.
- El ordenador compara que las claves simétricas para verificar que sean iguales.

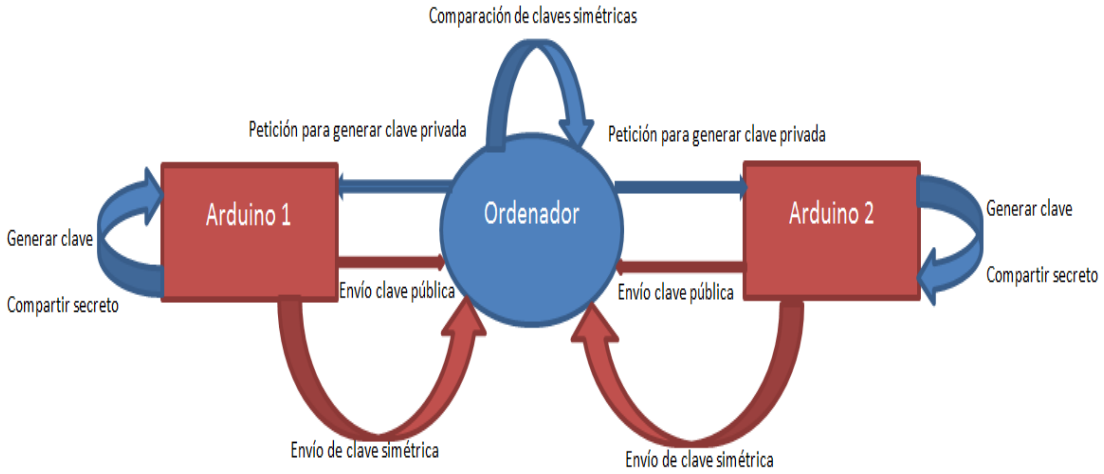

Figura 3.3: Diagrama de flujo de ECC.

3.2.4. XXTEA

El modelo del algoritmo de cifrado por bloques XXTEA es el mismo que el modelo del algoritmo AES-256.

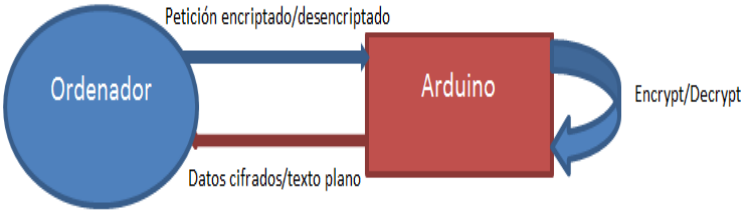

Figura 3.4: Diagrama de flujo de XXTEA.

4

Desarrollo

En este capítulo se va a describir la implementación del sistema desarrollado. Antes de profundizar en el desarrollo del sistema se va a especificar el software, el hardware, los entornos y los programas utilizados.

4.1. Hardware

4.1.1. Arduino Mega

Para el desarrollo de este proyecto se ha utilizado la placa Arduino Mega, la cual se basa en el MCU ATmega1280. Esta placa dispone de 54 pines digitales entrada/salida (14 salidas PWM), 16 entradas analógicas, 4 Universal Asynchronous Receiver-Transmitter (UART), un cristal de 16MHz, una conexión USB, un conector de alimentación, In-Circuit Serial Programming (ICSP) y un botón de reinicio. Mega es compatible con la mayoría de los shield para Diecimila y tiene una longitud y una anchura de 4 y 2,1 pulgadas.

Las características de Arduino Mega son las siguientes:

- MCU: ATmega1280.
- Tensión de funcionamiento: 5V.
- Voltaje de entrada recomendado: 7-12V.
- Voltaje de entrada sin límites: 6-20V.

- E/S digitales prendedores: 54, de los cuales 15 proporcionan salida PWM.
- Pines de entrada analógica: 16.
- Corriente continua para Pin I/O: 40 mA.
- Corriente continua para Pin 3.3V: 50 mA.
- Memoria flash: 128 KB de los cuales 4 KB son utilizados por el gestor de arranque.
- SRAM: 8KB.
- EEPROM: 4KB.
- Velocidad de reloj: 16 MHz. [14]

En la Figura 4.1 se puede ver una imagen de las dos placas Arduino utilizadas.

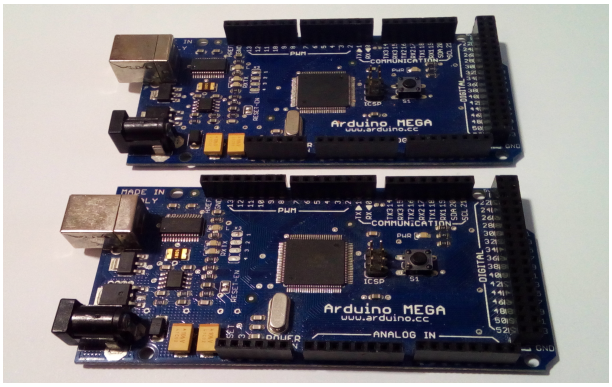

Figura 4.1: Arduino Mega usados en el proyecto.

4.1.2. MSI GE60 2PE Apache Pro

El ordenador usado para el desarrollo del proyecto es un MSI con las siguientes características:

- Procesador Intel Core i7-4700HQ, con 4 núcleos (8 hilos), una frecuencia de trabajo de 2,4-3,4 GHz y capaz de alcanzar una potencia máxima (Thermal Design Power (TDP)) de 47 W.
- RAM de 8 GB DDR3L.
- Sistema operativo Windows 7. [15]

4.2. Plataformas

Las plataformas utilizadas para el desarrollo del proyecto son las siguientes.

- **Ubuntu en Oracle VM VirtualBox**: es un software de visualización para arquitecturas x86/amd64, soporta sistemas operativos como GNU/Linux, Mac OS X, Microsoft Windows o MS-DOS. En este proyecto se ha instalado Ubuntu para probar algunos de los códigos implementados para Arduino.[16]

- **Sublime Text 2**: es un editor de textos y de código fuente, escrito en C++ y Python, el cual es usado para realizar la modificación de los códigos de los algoritmos criptográficos implementados.

- **Arduino 1.6.9**: IDE que permite el desarrollo de proyectos para Arduino. En este programa incluiremos las librerías que contienen los algoritmos criptográficos desarrollados. [14]

- **Eclipse Jee Mars**: IDE que contiene workspace base y un administrador de plugin que permite personalizar el entorno. Está escrito en Java y permite desarrollar aplicaciones en diferentes lenguajes de programación. Esta plataforma se ha utilizado para desarrollar la interacción de las placas Arduino con el ordenador, para ello se ha utilizado la librería RxTx mediante la cual se establecía la comunicación por el puerto serie. [17] [18]

- **Netbeans**: es un IDE de desarrollo integrado libre. Es un proyecto de código abierto muy popular y con un gran número de usuarios. Este entorno de programación ha sido utilizado en el desarrollo de los algoritmos criptográficos en C++.

4.3. Configuraciones realizadas

En este apartado se detallarán las configuraciones que han sido necesarias para obtener un correcto funcionamiento de los dispositivos utilizados.

4.3.1. Instalación del IDE de Arduino

Primero de todo se debe instalar el IDE de Arduino. El software de Arduino será necesario para poder cargar las librerías dentro de la placa Arduino. Esta instalación también creará una carpeta "C:/Program Files (x86)/Arduino/drivers" que contendrá el controlador de la placa Arduino.

Tras la instalación del software, se deberá conectar la placa Arduino e instalar el controlador del dispositivo. Para tal propósito, se deberá acceder al administrador de

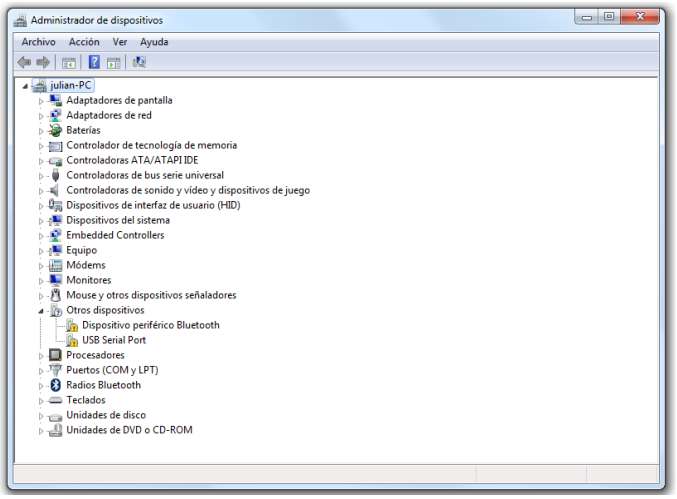

Figura 4.2: Dispositivo Arduino no reconocido.

dispositivos que se encuentra dentro del panel de control del ordenador. Una vez allí, encontraremos una imagen similar a la de la Figura 4.2.

Como se puede observar, inicialmente el ordenador no detecta correctamente la placa Arduino, esta aparecerá con una exclamación amarilla, lo que quiere decir que no tiene configurado ningún controlador para este dispositivo.

Para instalar el controlador, será necesario pinchar con el botón derecho del ratón sobre el nombre del dispositivo y seleccionar la opción "Actualizar software de controlador". Una vez hecho esto, aparecerá una ventana similar a la de la Figura 4.3, donde habrá que elegir si se quiere que el propio sistema de Windows busque el software del controlador en internet o en una carpeta específica.

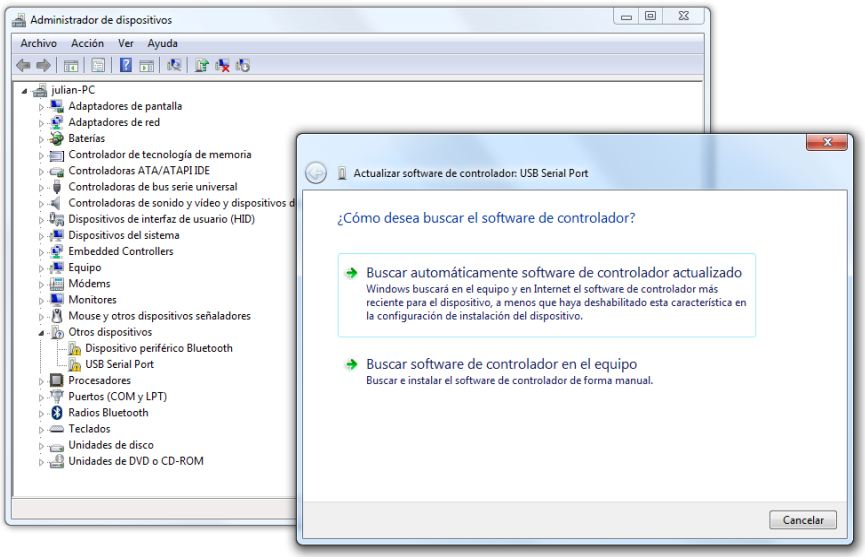

Figura 4.3: Ventana de búsqueda del controlador.

Se deberá seleccionar la opción "Buscar software de controlador en el equipo" e introducir la ruta de la carpeta que contiene el controlador, "C:/Program Files (x86)/Arduino/drivers". Una vez hecho esto, se pulsará el botón "Siguiente" y aparecerá una ventana similar a la de la figura Figura 4.4. Esta es una ventana de seguridad que muestra Windows para cerciorarse de que el usuario quiere instalar un controlador que no ha sido comprobado por el propio sistema operativo. Se confirmará la acción y, tras unos minutos, la placa Arduino será detectada correctamente por el ordenador y se le habrá asignado un puerto serie.

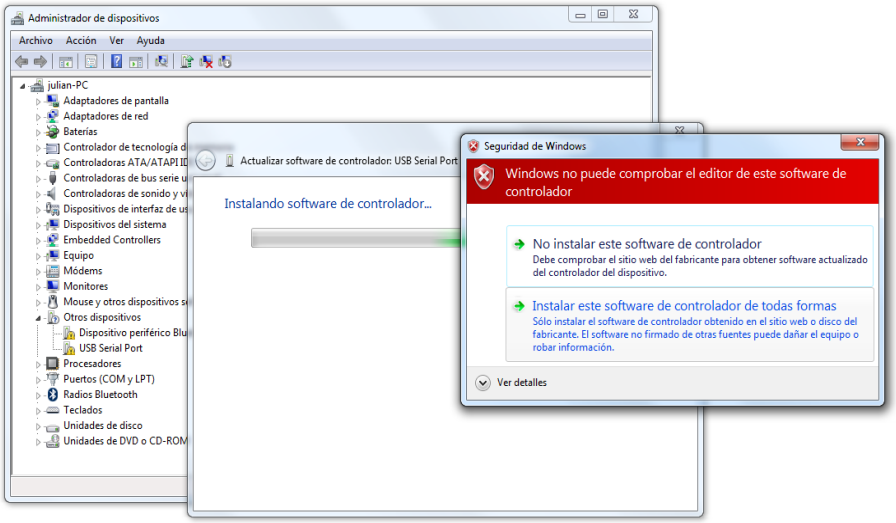

Figura 4.4: Ventana de seguridad.

4.3.2. Configuración del IDE de Arduino

Una vez instalado el IDE de Arduino e instalado el software del controlador de la placa, se deben realizar unas serie de configuraciones que van a permitir poder trabajar correctamente con Arduino. Estas configuraciones están relacionadas con el establecimiento de la conexión y van a permitir que el programa pueda comunicarse correctamente con la placa Arduino para subirle los programas implementados y depurarlos.

Primero de todo, hay que seleccionar la placa Arduino que se va a conectar. Para ello, en la parte superior del IDE, hay que dirigirse a herramientas y en el apartado placas seleccionar la placa Arduino con la que se esté trabajando. En este proyecto la placa Arduino utilizada es "Arduino/Genuino Mega or Mega 2560" (ver Figura 4.5).

En segundo lugar, se deberá seleccionar el procesador que posee la placa Arduino y el puerto COM al que está conectada la placa. Para ello, habrá que dirigirse de nuevo a herramientas y en el apartado procesador seleccionar el correspondiente a la placa que se está usando. Como puede observarse en la Figura 4.6, el MCU seleccionado ha sido el

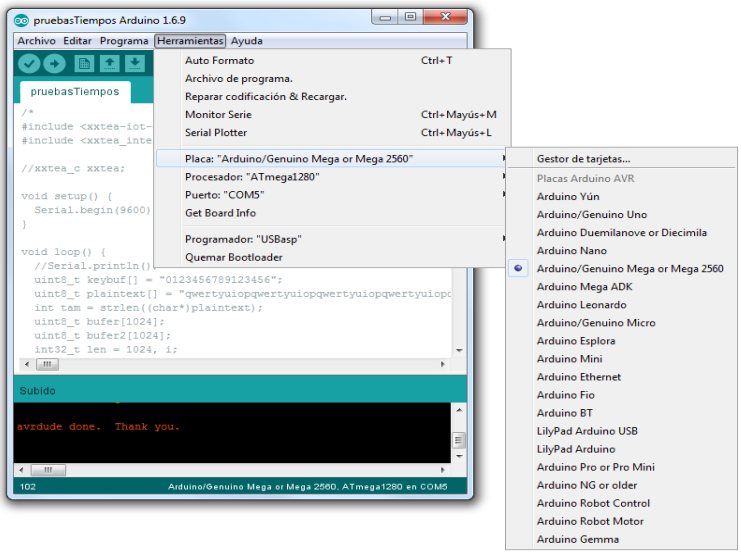

Figura 4.5: Configuración de la placa Arduino.

"ATmega1280", ya que es el MCU que utiliza la placa Arduino Mega, y el puerto COM donde se ha conectado la placa Arduino es el COM5.

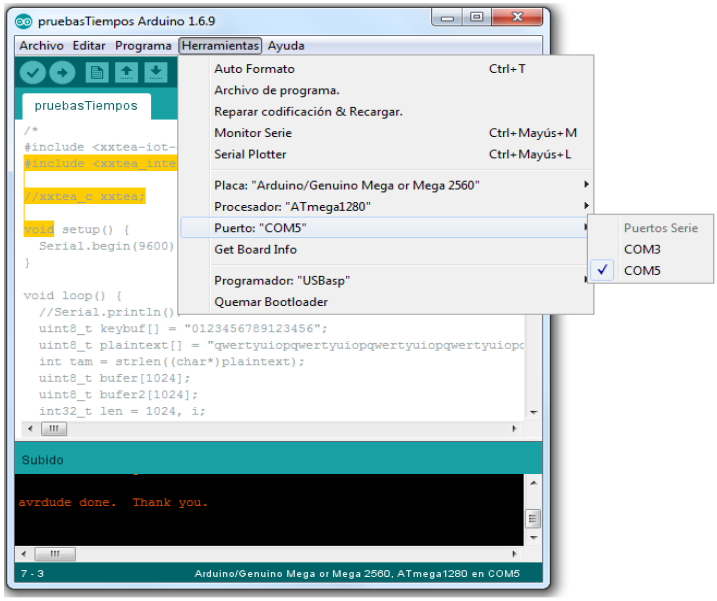

Figura 4.6: Configuración del MCU y el puerto COM.

Por último, se deberá seleccionar el puerto COM al que está conectada la placa Arduino (ver Figura 4.6).

4.4. Variaciones en la implementación

En esta sección se explicarán los cambios que han sido necesarios en la implementación de los algoritmos criptográficos.

En el caso del algoritmo ECC, se comenzó con la programación de la comunicación entre el ordenador y las dos placas Arduino en Linux. Para tal propósito, se usó la librería Messenger. Tras estudiar el funcionamiento de la librería, se observó que la librería RxTx, que se encontraba en el IDE Eclipse, simplificaba sustancialmente la comunicación. [19]

4.5. Librerías utilizadas

En esta sección se explicará el funcionamiento de la librería que ha hecho posible la comunicación entre Arduino y el ordenador (a través del IDE Eclipse).

La comunicación entre el ordenador y la placa Arduino se ha realizado mediante una conexión por puerto serie. Para poder establecer la conexión y realizar el intercambio de información correctamente se ha usado la librería RxTx de java. Esta librería se utiliza para enviar y recibir información a través del puerto serie.

4.6. Librerías implementadas

En esta sección se detallarán las librerías que han sido implementadas en el proyecto.

4.6.1. ShArduino

La librería de algoritmos hash posee una función principal llamada hash. Esta función es la encargada de realizar los tres pasos principales de un algoritmo hash. Estos pasos son:

- **Init**: inicializa un contexto incremental para cifrar.
- **Update**: carga datos en un contexto incremental de cifrado activo.
- **End**: finaliza un contexto incremental y devuelve el resultado cifrado. [20]

4.6.2. AesArduino

La librería del algoritmo simétrico AES contiene dos funciones principales encrypt y decrypt. En ambas funciones el funcionamiento y los pasos que se realizan son los mismos.

- **KeyExpansion**: esta función, a partir de la clave, genera tantos bytes como subclaves se necesitan para el algoritmo.

- **ByteSub**: esta función sustituye cada byte generando un byte nuevo.

- **ShiftRows**: esta función realiza un número de rotaciones hacia la izquierda de cada una de las filas de la matriz Estado, donde el número de rotaciones depende del número de la fila de esa matriz.

- **MixColumns**: esta función se encarga de hacer cambios en las columnas de la matriz Estado intermedio en la que se encuentre, realizando una multiplicación de las columnas por un polinomio que sea coprimo con cada columna.

- **AddRoundKey**: realiza una or-exclusive entre cada bit de la matriz Estado intermedia en la que se encuentre y la subclave generada a partir de la clave inicial de cifrado. El resultado de esto es una nueva matriz Estado. [21]

4.6.3. XXTeaArduino

La librería del algoritmo simétrico XXTEA contiene dos funciones principales encrypt y decrypt. En ambas funciones se realizan los mismos pasos: primero de todo se lee la clave de cifrado o descifrado, tras esto, se aplica el algoritmo Feitsel, realizando un número de ejecuciones que depende del tamaño del mensaje que se desea cifrar/descifrar.

4.6.4. EccArduino

En la librería del algoritmo criptográfico ECC destacan dos funciones:

- **MadeKey**: función que genera el par de claves privada/pública.

- **SharedSecret**: función que genera la clave simétrica.

No se pueden pasar por alto las funciones que calculan las curvas elípticas estándar, secp160r1, secp192r1, secp224r1, secp256r1, and secp256k1.

4.7. Algoritmos criptográficos

En esta sección se explicarán los prototipos implementados para cada uno de los algoritmos criptográficos desarrollados.

4.7.1. SHA-2

char * hash (char * buf, int size, int tipoSha);

Entradas:

- Buf: cadena del cual deseas generar el hash.
- Size: longitud del string buf.
- TipoSha: algoritmo hash que deseas realizar (256 o 512).

Salida:

- hash generado.

4.7.2. AES-128

void encrypt (uint8_t * output, uint8_t * input, uint32_t length, const uint8_t * key, const uint8_t * iv);

void decrypt (uint8_t * output, uint8_t * input, uint32_t length, const uint8_t * key, const uint8_t * iv);

Entradas:

- Output: cadena en el que se almacenará el resultado.
- Input: cadena que se desea cifrar/descifrar.
- Length: longitud de la cadena de entrada.
- Key: clave con la que se realizará la cifrado/descifrado.
- Iv: vector de iniciación.

Salida:

- Output: cadena en la que se almacenará el resultado.

4.7.3. XXTEA

void encrypt (uint8_t * output, uint8_t *input, int32_t len, uint8_t * key, int32_t * maxlen);

Entradas:

- Output: cadena en el que se almacenará el resultado.
- Input: cadena que se desea cifrar.
- Len: longitud de la cadena de entrada.
- Key: clave con la que se realizará el cifrado.
- Maxlen: máxima longitud de datos de entrada.

Salida:

- Output: cadena en la que se almacenará el resultado.

void decrypt (uint8_t * output, uint8_t * input, uint32_t len, uint8_t * key);

Entradas:

- Output: cadena en el que se almacenará el resultado.
- Input: cadena que se desea cifrar.
- Length: longitud de la cadena de entrada.
- Key: clave con la que se realizará el descifrado.

Salida:

- Output: cadena en la que se almacenará el resultado.

4.7.4. ECC

void uECC_set_rng (uECC_RNG_Function rng_function);

Entradas:

- Función de generación de número aleatorio.

int uECC_make_key (uint8_t * public_key, uint8_t * private_key, uECC_Curve curve);

Entradas:

- Public_key: cadena donde se almacenará la clave pública.
- Private_key: clave privada propia.
- Curve: curva elíptica.

Salida:

- Output: clave pública generada.
- Entero para comprobar la correcta realización de la clave.

int uECC_shared_secret (const uint8_t * public_key, const uint8_t * private_key, uint8_t * secret, uECC_Curve curve);

Entradas:

- Public_key: clave pública del otro Arduino.
- Private_key: clave privada propia.
- Secret: cadena donde se almacenará la clave simétrica generada.
- Curve: curva elíptica.

Salida:

- Secret: clave simétrica generada.
- Entero para comprobar la correcta realización de la clave simétrica.

5

Integración, pruebas y resultados

En esta sección se abordarán los procesos de integración y las pruebas realizadas durante y tras el desarrollo del proyecto. El objetivo de las pruebas es la detección y corrección de errores antes de la finalización y presentación del proyecto.

5.1. Pruebas de usabilidad

Las pruebas de usabilidad consisten en la selección de un grupo de usuarios a los cuales se les solicitará que lleven a cabo una serie de pruebas para comprobar el correcto funcionamiento de la aplicación mientras que los desarrolladores toman notas de los resultados de dichas pruebas. Varios estudiantes de la Universidad Autónoma de Madrid han podido comprobar el correcto funcionamiento de los algoritmos criptográficos implementados. [22]

5.2. Pruebas de compatibilidad

Las pruebas de compatibilidad se emplean para asegurarse de que el software desarrollado sea compatible en distintos sistemas operativos, hardware y otros. Inicialmente el código de los algoritmos desarrollados se compiló y ejecutó en Linux y, tras comprobar su correcto funcionamiento, se trasladó a Arduino, realizando ligeros cambios en el código, donde se repitió el mismo proceso, verificando así que funcionaba en ambos casos. [22]

5.3. Pruebas unitarias

Las pruebas unitarias permiten comprobar el correcto desempeño de la funcionalidad de cada uno de los módulos por separado. [22]

Función a comprobar	Descripción	Resultado
SHA-256	Se introduce una cadena de 512 bits para realizar el hash.	Correcto. Se genera el hash bien.
SHA-512	Se introduce una cadena de 1024 bits para realizar el hash.	Correcto. Se genera el hash bien.
Encript AES-128	Se introduce una cadena de 512 bits para realizar el cifrado.	Correcto. Se genera la cadena cifrada bien.
Encript AES-128	Se introduce una cadena de 1024 bits para realizar el cifrado.	Correcto. Se genera la cadena cifrada bien.
Decrypt AES-128	Se introduce una cadena de 512 bits para realizar el descifrado.	Correcto. Se genera la cadena descifrada bien.
Decrypt AES-128	Se introduce una cadena de 1024 bits.	Correcto. Se genera la cadena descifrada bien.
Encrypt XXTEA	Se introduce una cadena de 512 bits para realizar el cifrado.	Correcto. Se genera la cadena cifrada bien.
Decrypt XXTEA	Se introduce una cadena de 1024 bits para realizar el descifrado.	Correcto. Se genera la cadena descifrada bien.
ECC compress	Comprime la clave pública dependiendo del tamaño de la curva más uno.	Correcto. Clave pública comprimida bien.
ECC decompress	Descomprime la clave pública comprimida.	Correcto. Clave pública comprimida descomprimida bien.
ECC compute	Calcula la clave pública a través de la clave privada.	Correcto. Clave pública calcula bien.
ECC shared secret	Calcula la clave simetrica usando su clave privada y la clave pública de cualquier otro.	Correcto. Se genera la clave simétrica bien.
ECC verify	Comprueba una firma ECDH.	Correcto. La comprobación es afirmativa.

Función a comprobar	Descripción	Resultado
ECC sign	Calcula una firma ECDH para un determinado hash.	Correcto. Se genera el hash bien.
ECC sign deterministic	Calcula una firma ECDH para un determinado hash, utilizando un algoritmo determinista.	Correcto. Se genera el hash bien.
ECC make key	Genera un par de claves pública/privada.	Correcto. Se genera la clave pública bien y la clave privada.

Tabla 5.1: Pruebas unitarias realizadas.

5.4. Pruebas de caja negra

Las pruebas de caja negra permiten observar el algoritmo de encriptación desde el punto de vista de las entradas recibidas y las salidas generadas. En este proyecto se ha comprobado que la cadena de entrada que se desea cifrar sea la misma que la cadena obtenida tras el descifrado o, en el caso de ECC, que las claves simétricas generadas por las dos placas Arduino sean iguales. [22]

5.5. Pruebas de integración

Las pruebas de integración se realizarán una vez se hayan llevado a cabo las pruebas unitarias y el sistema o subsistema se haya finalizado, puesto que en ellas se comprobará tanto el adecuado funcionamiento de cada módulo individualmente como su correcta interactuación con el resto de módulos. En la siguiente tabla se muestran las pruebas de integración de ECC. [22]

Componentes a integrar	Descripción	Resultado
Arduino-1 y ordenador.	Envío de clave pública de Arduino-1 al ordenador.	Correcto. El ordenador recibe completa la clave pública-1.
Arduino-2 y ordenador.	Envío de clave pública de Arduino-2 al ordenador.	Correcto. El ordenador recibe completa la clave pública-1.
Ordenador y Arduino-1.	Envío de clave pública (del Arduino-2) de ordenador al Arduino-1.	Correcto. La placa Arduino-2 recibe completa la clave pública-1.

Componentes a integrar	Descripción	Resultado
Ordenador y Arduino-2.	Envío de clave pública (del Arduino-2) de ordenador al Arduino-1.	Correcto. La placa Arduino-1 recibe completa la clave pública-2.
Arduino-1 y ordenador.	Envío de clave simétrica de Arduino-1 al ordenador.	Correcto. El ordenador recibe completa la clave simétrica-1.
Arduino-2 y ordenador.	Envío de clave simétrica de Arduino-2 al ordenador.	Correcto. El ordenador recibe completa la clave simétrica-2.
Ordenador.	El ordenador compara las dos claves simétricas para ver que el resultado sea el mismo.	Correcto. La clave simétrica-1 es igual a la clave simétrica-2.

Tabla 5.2: Pruebas de integración realizadas.

5.6. Muestra de ejecución ECC

En esta sección se explicará como sería una ejecución del algoritmo ECC.

Lo primero que aparece es el menú inicial, se deberá pulsar uno para que el ordenador envíe la petición de generación del par de clave privada/pública a las dos placas Arduino (Figura 5.1).

```
MENU
1.Make key
2.Enviar contrasenha publica
3.Shared secret
Enter your choice :
1
```

Figura 5.1: Menu ECC.

Tras el envío de la petición, el ordenador recibirá e imprimirá las claves públicas de las dos placas Arduino (Figura 5.2).

```
Sent: 1
57   97   19   127   -121   -121   -69   13   -46   -27   20
61   -114   45   -67   -107   -123   41   74   -13   29   -67   28   -51   37   -46
-94   60   -37   -65   53   -55   126   115   120   92   122   34   -57   5
Sent: 1
-3
-109   -6   -68   52   60   -123   -10   80   -51   -64   -71   74   -77   -98   71   112
-113   65   -87   56   -57   37   -81   -39   48   -75   79   -63   42   -28   93
-4   102   96   49   -117   53   -127   -48
```

Figura 5.2: Salida en las placas Arduino de las claves públicas generadas.

Tras recibir las dos claves públicas, se debe enviar un dos para enviar la clave pública de cada Arduino a la placa Arduino del otro extremo (Figura 5.3).

```
1.Make key
2.Enviar contrasenha publica
3.Shared secret
Enter your choice :
2
25   42   -44   93   -128   96   -90   -22   81   -58   17   -109   52   111   99   32   -41   -52   115   63
-36   59   54   -83   -124   105   111   47   -31   16   24   -20   -84   -53   89   -85   -37   -75   90   -127
Sent: 2
Sent: [B@61064425
101   -17   10   60   9   -33   106   -17   -57   53   -107   -63   -3   89   59   -12   41   74   -4   8
49   14   -24   -14   -78   -97   15   87   96   20   -111   -27   -24   -18   -105   -28   -86   -100   21   -47
Sent: 2
Sent: [B@7b1d7fff
```

Figura 5.3: Claves públicas recibidas de las placas Arduino.

Por último el ordenador recibe las claves simétricas de cada uno de los Arduino, de esta manera se puede comprobar que ambas claves públicas son iguales y que el funcionamiento del algoritmo ECC es el correcto (Figura 5.4).

```
MENU
1.Make key
2.Enviar contrasenha publica
3.Shared secret
Enter your choice :
3
Sent: 3
64   -100   -86
25   -98   12   21   85   54   23   5   -107   28   -104   -94   86   -49   -121   -96
93
Sent: 3
64   -100   -86   25   -98   12   21   85   54   23
5   -107   28   -104   -94   86   -49   -121   -96   93
```

Figura 5.4: Clave simétrica generada por las placas Arduino.

6

Conclusiones y trabajo futuro

En esta sección se mostrarán los rendimientos obtenidos tras la ejecución de cada uno de los algoritmos implementados en este proyecto.

6.1. Conclusiones

Para la obtención de los datos del rendimiento se ha seguido el siguiente procedimiento en cada uno de los algoritmos:

- Ejecución del algoritmo 10 veces e impresión de los resultados de los tiempos en un fichero.
- Eliminación del mayor y el menor resultado.
- Cálculo de la media.

Para la obtención de los valores de tiempo se ha utilizado la función millis() de la librería de Arduino. Esta función se colocaba antes y después de la llamada al algoritmo para obtener los timpos de inicio y fin de la ejecución. Una vez se tenían estos datos, se realizaba una resta, TiempoDeInicio - TiempoDeFin para obtener el tiempo total de ejecución.

Finalmente los resultados obtenidos para cada uno de los algoritmos, tanto en la placa Arduino como en el PC, son los que se muestran en la Tabla 6.1, la Tabla 6.2, la Tabla 6.3 y la Tabla 6.4.

Algoritmo	Datos obtenidos(ms)	Media aritmética(ms)	Varianza(ms)
SHA-256	97, 97, 97, 97, 97, 97, 97, 98	97.125	0.109
SHA-512	81, 81, 81, 81, 81, 81, 81, 81	81	0
AES-256 encrypt	27, 28, 28, 28, 28, 28, 28, 28	27.875	0.109
AES-256 decrypt	28, 29, 28, 28, 29, 28, 29, 28	28.375	0.234
XXTEA encrypt	15, 15, 15, 15, 15, 16, 16, 16	15.375	0.234
XXTEA decrypt	14, 14, 14, 15, 15, 15, 15, 15	14.625	0.234

Tabla 6.1: Tiempos de algoritmos en Arduino para una cadena de 512 bits.

Algoritmo	Datos obtenidos(ms)	Media aritmética(ms)	Varianza(ms)
SHA-256	0.056, 0.057, 0.054, 0.055, 0.056, 0.056, 0.056, 0.056	0.056	6.875E-7
SHA-512	0.054, 0.054, 0.055, 0.055, 0.055, 0.055, 0.056, 0.056	0.055	5E-7
AES-256 encrypt	0.311, 0.315, 0.315, 0.317, 0.322, 0.327, 0.328, 0.329	0.321	4.2E-5
AES-256 decrypt	0.305, 0.309, 0.315, 0.322, 0.327, 0.328, 0.329, 0.349	0.323	1.672E-3
XXTEA encrypt	0.015, 0.009, 0.009, 0.009, 0.009, 0.009, 0.009, 0.013	1.025E-2	4.938E-6
XXTEA decrypt	0.01, 0.012,0.009, 0.009, 0.009, 0.009, 0.009, 0.009	0.0095	1.0E-6

Tabla 6.2: Tiempos de algoritmos en el ordenador para una cadena de 512 bits.

Algoritmo	Datos obtenidos(ms)	Media aritmética(ms)	Varianza(ms)
SHA-256	184, 184, 184, 184, 184, 184, 184, 185	184.125	0.109
SHA-512	144, 144, 144, 145, 145, 145, 145, 145	144.625	0.234
AES-256 encrypt	56, 55, 55, 56, 56, 56, 55, 55	55.5	0.25
AES-256 decrypt	55, 55, 56, 55, 55, 55, 55, 55	55.125	0.109
XXTEA encrypt	29, 29, 29, 30, 30, 30, 30, 30	29.625	0.234

Algoritmo	Datos obtenidos(ms)	Media aritmética(ms)	Varianza(ms)
XXTEA decrypt	26, 27, 27, 27, 27, 27, 27, 27	26.875	0.109

Tabla 6.3: Tiempos de algoritmos en Arduino para una cadena de 1024 bits.

Algoritmo	Datos obtenidos(ms)	Media aritmética(ms)	Varianza(ms)
SHA-256	0.016, 0.017,0.017, 0.019, 0.022,0.022, 0.029, 0.031	0.022	2.798E-5
SHA-512	0.02,0.02, 0.021, 0.023, 0.029, 0.034, 0.037, 0.04	0.028	5.8E-5
AES-128 encrypt	0.741, 0.743, 0.75, 0.751, 0.756, 0.764, 0.768, 0.782	756875	1.666E-4
AES-128 decrypt	0.723, 0.741, 0.743, 0.748, 0.75, 0.759, 0.764, 0.771	0.749875	1.976E-4
XXTEA encrypt	0.027, 0.018, 0.018, 0.018, 0.018, 0.018, 0.018, 0.018	0.0191	8.859E-6
XXTEA decrypt	0.019, 0.018, 0.018, 0.018, 0.018, 0.018, 0.018, 0.019	0.018	1.875E-7

Tabla 6.4: Tiempos de algoritmos en el ordenador para una cadena de 1024 bits.

En el desarrollo de este trabajo se ha intentado implementar el algoritmo RSA pero no ha sido posible. RSA es un algoritmo criptográfico de clave pública basado en la factorización de números grandes. Actualmente es el algoritmo asimétrico más utilizado. El algoritmo RSA utiliza muchos recursos ya que debe calcular y trabajar con números primos muy grandes y utilizar funciones recursivas, para lo que necesita grandes cantidades de memoria, cantidades que no posee Arduino. Este ha sido el motivo por el cual no se ha podido implementar satisfactoriamente el algoritmo. Es por esto que se decidió implementar el algoritmo ECC, quería comprobarse si este algoritmo poseía los mismos problemas que RSA con la memoria de Arduino. Durante el desarrollo del algoritmo ECC no se encontraron problemas relacionados con la memoria y pudo ejecutarse satisfactoriamente obteniendo los tiempos que se muestran en la Tabla 6.5 y la Tabla 6.6. [23]

Algoritmo	Datos obtenidos(ms)	Media aritmética(ms)	Varianza(ms)
Make key public/private	1095, 1096, 1099, 1174, 1178, 1184, 1192 ,1243	1157,625	2625,7344
Make key simetric	1225, 1236, 1247, 1249, 1245, 1245, 1253, 1257	1244,625	88,4844

Tabla 6.5: Tiempos del algoritmo ECC en Arduino.

Algoritmo	Datos obtenidos(ms)	Media aritmética(ms)	Varianza(ms)
Make key public/private	7.885, 9.737, 7.461, 8.378, 9.348, 9.08, 12.356, 12.486	9.591	3.161
Make key simetric	6.245, 7.929, 4.465, 6.587, 7.369, 7.732, 8.905, 6.305	6.942	1.598

Tabla 6.6: Tiempos del algoritmo ECC en el ordenador.

6.2. Trabajo futuro

Para finalizar, en este apartado se definen las líneas futuras que seguir tomar el proyecto:

- Desarrollo del algoritmo RSA para la placa Arduino. Se pretende comparar el rendimiento del algoritmo RSA con el algoritmo ECC y demostrar que algoritmo ECC es menos pesado y más rápido, computacionalmente hablando.
- Desarrollo de otros algoritmos criptográficos, como por ejemplo el algoritmo Digital Signature Algorithm (DSA).
- Verificar el correcto funcionamiento de los algoritmos implementados en otro tipo de placa Arduino diferente a la placa Arduino Mega.
- Conseguir que las placas Arduino que implementan los algoritmos vistos en el proyecto puedan ser llevadas a sistemas reales. De esta manera, se pretende verificar que el funcionamiento de los algoritmos es correcto y que el rendimiento es lo suficientemente bueno como para poder llegar a implantar placas Arduino que implementen algoritmos criptográficos dentro de este tipo de sistemas.

Bibliografía

[1] *Ciberseguridad*. 2010. URL: http://www.itu.int/net/itunews/issues/2010/09/20-es.aspx.

[2] *Los pedidos mundiales de microcontroladores para el año 2017*. 2013. URL: http://www.convertronic.net/Informes/los-pedidos-mundiales-de-microcontroladores-de-seguridad-cibernetica-aumentaran-un-91-por-ciento-para-el-ano-2017.html.

[3] *Historia de Arduino y su nacimiento*. 2012. URL: https://botscience.wordpress.com/2012/06/05/historia-de-arduino-y-su-nacimiento/.

[4] *Microcontroladores*. 2013. URL: http://perso.wanadoo.es/pictob/microcr.htm.

[5] *Quakescape 3D Fabricator*. 2015. URL: http://jamesboock.com/Quakescape-3D-Fabricator.

[6] Booby Tisoy. *http://latestarduinoprojects.blogspot.com.es/2014/09/the-eyewriter-20-using-arduino-full.htm*. 2016.

[7] *13 proyectos asombrosos con Arduino*. 2015. URL: http://www.xataka.com/makers/13-proyectos-asombrosos-con-arduino-para-ponerte-a-prueba-y-pasar-un-gran-rato.

[8] Isaac PE. *http://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/*. 2014.

[9] Alfred J. Menezes y col. *Handbook of Applied Cryptography*. CRC Press, 1997.

[10] Bruce Schneier. *Applied Cryptography – Protocols, Algorithms, and Source Code in C*. John Wiley y Sons.

[11] Susan F. Marseken Lambert M. Surhone Miriam T. Timpledon. *Tiny Encryption Algorithm*. Betascript Publishers, 2010.

[12] Tadayoshi Kohno Niels Ferguson Bruce Schneier. *Cryptography Engineering: Design Principles and Practical Applications*. John Wiley y Sons Ltd, March 2010.

[13] Menezes Alfred J. Vanstone Scott Hankerson Darrel. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[14] *Arduino*. 2016. URL: https://www.arduino.cc/en/Main/ArduinoBoardMega2560.

[15] *MSI*. 2016. URL: https://es.msi.com/Laptop/GE60-2PE-Apache-Pro.html#hero-overview.

[16] *VirtualBox*. 2016. URL: https://www.virtualbox.org/.

[17] *Eclipse*. 2016. URL: http://www.eclipse.org/.

[18] *Open source software*. 2016. URL: http://fizzed.com/oss/rxtx-for-java.

[19] *USE THIS INSTEAD: CmdMessenger*. 2016. URL: http://playground.arduino.cc/Code/Messenger.

[20] *crypto*. 2016. URL: http://erlang.org/doc/man/crypto.html.

[21] *Cifrado con algoritmo AES*. 2013. URL: http://www.dmdcosillas.org/2014/08/cifrado-de-particiones-algoritmo-aes-i/.

[22] Roger Pressman. *Ingeniería del software un enfoque práctico*. 2010.

[23] *¿Qué es RSA?* 2007. URL: https://seguinfo.wordpress.com/2007/09/14/%C2%BFque-es-rsa/.